REMARKS/ARGUMENTS

1. Summary of the Office Action

Claims 7 and 15 are cancelled.

Claims 1-6, 8-14, and 16-18 are pending. Claims 1-6, 8, 17, and 18 stand rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Dzikewich et al., U.S. Patent No. 5,706,500 ("Dzikewich") in view of Horiguchi et al., U.S. Patent No. 6,073,157 ("Horiguchi").

Claims 9-14, and 16 stand rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Dzikewich in view of Matsuda et al., U.S. Patent No. 5,790,419 ("Matsuda").

2. Response to § 103 Rejections

Applicantrespectfully traverses this rejection for the reasons set out below, and ask the Examiner for reconsideration.

To establish a **prima facie** case of **obviousness**, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure. In re Vaeck, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).

THE PRIOR ART REFERENCES DO NOT TEACH OR SUGGEST ALL CLAIM LIMITATIONS, WHEN CONSIDERED SINGULARLY OR IN COMBINATION.

Claim 1, as amended, includes the following limitations:

automatically detecting an exit of a child application object, wherein a grandchild application object launched by the child application and a parent application object that launched the child application object remain extant;

automatically terminating the grandchild application object after the exit of the child application;

automatically determining whether the exit of the child application object was expected;

automatically attempting restart of the child application object if the exit of the child application object was unexpected; and

automatically signaling an outcome of the restart to the parent application object that launched the child application object. (emphasis added)

The teachings of Dzikewich in combination with Horiguchi fail to render the present claims obvious because these references, individually or in combination, simply do not teach or suggest automatically detecting an exit of a child application object, wherein a grandchild application object launched by the child application object and a parent application object that launched the child application object remain extant, automatically terminating the grandchild application object after the exit of the child application, automatically determining whether the exit of the child application object was expected, automatically attempting restart of the child application object if the exit was

unexpected, and automatically signaling an outcome of the restart to the parent application object that launched the child application object.

Dzikewich teaches the operation of a transaction system including a business event processor configured to start or restart appropriate scripts for processing data. Each unit of work includes an application context and a script processing state. As illustrated by Figure 3A and correspondingly explained in the specification, when the transaction system has been restarted, the business event processor checks if the application context and script processing states are available in the restart data sets. If the data sets are available, the business event processor rebuilds the application and if no data sets are available, then all previously run scripts completed successfully (C4, L5-22). As Figure 3B illustrates and is correspondingly described in the detailed description, in step 9, the business event processor stores (or deletes if the last unit of work processed) the application context and the script processing state into (or from) a recoverable storage. Thus, the application context and the script processing state are stored in memory, and are only relevant in the event the whole system crashes, restarts and checks this memory location to determine where to begin processing units of work. Consequently, the system cannot detect an exit of a child application object because the system itself has exited and therefore can not possibly automatically detect an exit of a child application object, wherein a grandchild application object launched by the child application object and a parent application object that launched the child application object remain extant, as recited in claim 1.

Further, Dzikewich does not automatically determine whether the exit of the child application object was expected as recited in claim 1. Dzikewich merely discloses that, upon restart of the system, the business event processor checks if the application context and script processing states are available in the restart

data sets. The business event processor merely checks, rebuilds and restarts the script. As a result, Dzikewich does not automatically determining whether the exit of the child application object was expected, and particularly not while the grandchild application object launched by the child application object and a parent application object that launched the child application object remain extant, as recited in claim 1.

Adding the teachings of Horiguchi, alone or in combination with Dzikewich, fails render claim 1 as obvious since it does not disclose automatically detecting an exit of a child application object, wherein a grandchild application object launched by the child application object and a parent application object that launched the child application remain object extant, automatically terminating the grandchild application object after the exit of the child application, automatically determining whether the exit of the child application object was expected, automatically attempting restart of the child application object if the exit was unexpected, and automatically signaling an outcome of the restart to the parent application object that launched the child application object. Instead, Horiguchi discusses a method for managing computer program execution by managing the execution of processes that include one or more enclaves. An enclave is defined as a logical run-time structure that supports the execution of a group of procedures and can have multiple threads. A thread is an execution construct consisting of synchronous invocations and terminations of invocation units (including procedures). When an enclave terminates, all threads are terminated. Claim 1 presently recites automatically detecting an exit of a child application object, wherein a grandchild application object launched by the child application object, and a parent application object that launched the child application object, remain extant and automatically terminating the grandchild application object after the exit of the

child application object, and automatically determining whether the exit of the child application object was expected. Horiguchi merely discloses that when an enclave terminates, all threads are terminated (col 6, ln 47-49). There in not an indication or disclosure, inherent or otherwise, in Horiguchi that if an enclave (child) terminates that the thread (grandchild) and process (parent) will remain extant until a step of automatically terminating the grandchild application object after the exit of the child application object is performed. Further, Horiguchi only generally discusses termination of an enclave (child) and does not discuss or disclose automatically determining whether the exit of the child application object was expected as recited in claim 1.

Because independent claims 17 and 18 have substantially similar limitations as claim 1, the same arguments that applied to claim 1 also apply to claims 17 and 18. Therefore, for at least the reasons stated above, independent claims 1, 17 and 18, and all claims dependent thereon, are patentable over the cited art.

The teachings of Dzikewich in combination with Horiguchi and in further view of Matsuda fail to render the present claims obvious because, individually or in combination, neither teach nor suggest a watchdog automatically to detect an exit of a child application object and an executor automatically to terminate a grandchild application object launched by the child application object after the exit of the child application object, to automatically determining whether the exit of the child application object was expected, to automatically attempt restart of the child application object if the exit was unexpected, and to automatically signal an outcome of the restart to a parent application object that launched the child application object. Matsuda discusses a system for determining when a microcomputer is in an abnormal state. Specifically, Matsuda discloses a watchdog timer that monitors a pulse train frequency from a controller within

the microcomputer circuit. When the pulse train falls outside a specified frequency range, the microcomputer is in an abnormal state and the watchdog timer outputs a signal to the controller that causes the microcomputer to reset. Matsuda does not discuss, as in claim 9, a watchdog automatically to detect an exit of a child application object, but instead discusses a piece of hardware, a microcomputer, that is monitored by a watchdog that detects irregularities in a pulse train. Clearly, an exit, or failure, of a piece of hardware is not equivalent to the exit, or failure, of a child application.

Therefore, for at least the reasons stated above, independent claim 9 and all dependent claim limitations therefrom are patentable over the cited art.

THERE IS NO SUGGESTION OR MOTIVATION TO COMBINE THE REFERENCES.

To rely on a reference under 35 U.S.C. 103, the reference must be analogous prior art (MPEP 2141.01(a)). While Patent Office classification of references and cross-references in the official search notes are some evidence of "nonanalogy" or "analogy" respectively, the court has found "the similarities and differences in structure and function of the inventions to carry far greater weight." (In re Ellis, 476 F.2d 1370, 1372). The Applicant respectfully asserts that a process, enclave, and thread represent levels of descending abstraction that rely on each other to perform their respective functions, similar to a computer program, program code (e.g., FORTRAN), and machine language, respectively. The present claims recite parent, child and grandchild application objects without these levels of descending abstraction. In other words, the parent, child, grandchild application object relationship may be based upon which application object launched the other and may not otherwise be dependent upon each other for functional execution, as is the case with a

process, enclave and thread. Therefore, the structure and function of the "hierarchy" disclosed in Horiguchi is significantly different than that of claim 1 as presently recited and should be considered non-analogous art.

Additionally, the reference is not reasonably pertinent to the problem with which the Applicant was concerned because a person having ordinary skill in the art would not reasonably have expected to solve the problem of automatically terminating extant applications in a hierarchy when the calling application fails by considering a reference dealing a method for managing program execution that unifies widely disparate models presented by various high-level language (e.g., C, FORTRAN, COBOL, etc.) and operating system standards.

In light of the above, the Applicant respectfully submits that the rejection under 35 U.S.C. § 103 has been overcome, and withdrawal of this rejection is therefore respectfully requested.

3. Conclusion

Having tendered the above remarks and amended the claims as indicated herein, the Applicant respectfully submits that all rejections have been addressed and that the claims are now in a condition for allowance, which is earnestly solicited.

If there are any additional charges, please charge Deposit Account No. 02-2666. If a telephone interview would in any way expedite the prosecution of the present application, the Examiner is invited to contact André Marais at (408) 947-8200 ext. 204.

Respectfully submitted,
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Dated: 11/24/, 2003

André L. Marais
Reg. No. 48,095

12400 Wilshire Blvd.
Seventh Floor
Los Angeles, CA 90025-1026
(408) 947-8200